Claims

1. A data structure for representing control constraints of a software system, the software system comprising at least two software elements with explicit control interactions between the software elements, the data structure comprising:

a set of conjunctive nodes, in which each conjunctive node of the set of conjunctive nodes represents a conjunctive boolean guard on state changes within the software system;

a set of disjunctive nodes, in which each disjunctive node of the set of disjunctive nodes represents a boolean guard on a functional object within one of the software elements; and

a set of directed edges, in which each directed edge of the set of directed edges connects two nodes and represents implication between the nodes.

2. A data structure according to claim 1 wherein each directed edge of the set of directed edges has an origin and a destination and only responds to a true value at the origin.

3. A data structure according to claim 1 wherein each directed edge of the set of directed edges has an origin and a destination and only responds to a false value at the origin.

4. A data structure according to claim 1 wherein each directed edge of the set of directed has an origin and a destination and only asserts a false value at the destination.

5. A data structure according to claim 1 wherein each directed edge of the set of directed has an origin and a destination and only asserts a true value at the destination.

6. A data structure according to claim 1 wherein each directed edge of the set of directed edges has an origin, in which the edge responds to a value selected from the group consisting of true and false, at the origin.

7. A data structure according to claim 6 wherein each directed edge of the set of directed edges has a destination, in which the edge asserts a value selected from the group consisting of true and false, at the destination.

8. A software analysis tool for debugging a software system, the software system having software elements which expose control interactions between the software elements, by representing the control interactions in a graph, the graph comprising:

a set of conjunctive nodes, which represent boolean guards on state changes within a software element;

a set of disjunctive nodes, which represent boolean guards on a functional object within the software elements; and

a set of directed edges, which represent implication between the nodes.

9. A data structure according to claim 8 wherein each directed edge of the set of directed edges has an origin and a destination and only responds to a true value at the origin.

10. A data structure according to claim 8 wherein each directed edge of the set of directed edges has an origin and a destination and only responds to a false value at the origin.

11. A data structure according to claim 8 wherein each directed edge of the set of directed has an origin and a destination and only asserts a false value at the destination.

12. A data structure according to claim 8 wherein each directed edge of the set of directed has an origin and a destination and only asserts a true value at the destination.

13. A data structure according to claim 8 wherein each directed edge of the set of directed edges has an origin in which the edge responds to a value selected from the group consisting of true and false, at the origin.

14. A data structure according to claim 13 wherein each directed edge of the set of directed edges has a destination in which the edge asserts a value selected from the group consisting of true and false, at the destination.

15. A method of modifying a static control graph to support detecting constraint enforcement conflicts in a distributed software system modeled by the static control graph, the static control graph including a set of disjunctive nodes D, a set of conjunctive nodes C, and a set of edges E each representing an implication interrelating the nodes it contacts, and the method comprising the steps of:

selecting a first disjunctive node in the static control graph having a first incident edge from a first conjunctive node and having a first outgoing edge directed to a second conjunctive node;

checking whether the first incident edge and the first outgoing edge assert and respond to a consistent value;

proposing a new conjunctive node associated with the selected disjunctive node;

checking whether the proposed new conjunctive node is redundant of any existing conjunctive nodes in the static control graph; and

if the proposed new conjunctive node is consistent and unique, adding the new conjunctive node to the static control graph;

identifying a first set of disjunctive nodes which comprises every disjunctive node that provides an input edge to the first conjunctive node;

for each disjunctive node in the first set of disjunctive nodes, creating a corresponding sensing edge input from said disjunctive node in the first set of disjunctive nodes to the new conjunctive node;

identifying a second set of disjunctive nodes which comprises every disjunctive node other than the selected first disjunctive node that provides an input edge to the second conjunctive node;

for each disjunctive node in the second set of disjunctive nodes, creating a corresponding input edge from said disjunctive node in the second set of disjunctive nodes to the new conjunctive node;

identifying a third set of disjunctive nodes which comprises every disjunctive node that responds to an output edge from the second conjunctive node;

for each identified disjunctive node in the third set of disjunctive nodes, creating a corresponding enforcing edge from the new conjunctive node to the disjunctive node in the third set of disjunctive nodes;

identifying in the modified graph any pairs of edges that assert different values on a disjunctive node; and

determining whether the source conjunctive nodes of the identified pair of edges are mutually exclusive.

16. A method according to claim 15 and further comprising repeating the foregoing steps for every disjunctive node in the static control graph not previously selected that receives an input from a conjunctive node and generates an output to another conjuctive node, thereby exposing potential constraint conflicts in the design represented by the static control graph.

17. A method according to claim 16 wherein the method modifies a static control graph for each layer of a software system independently of every other layer of the software system, the software system having multiple layers in which interactions between the layers are through opaque actions.

18. A method according to claim 16 further comprising:

analyzing the static control graph;

tracing a potential constraint conflict to contributing nodes; and

identifying a set of node values that caused the potential constraint conflict.

19. A method of creating a static control graph to support detecting constraint enforcement conflicts in a software system, the software system having at least two software elements with explicit control interactions between the software elements, and the method comprising the steps of:

creating a new disjunctive node for each mode in the software system;

creating a new conjunctive node for each pair of corresponding constraints in the software system;

creating, for each new conjunctive node that generates an output value, a new outgoing edge from the new conjunctive node to a corresponding disjunctive node; and

creating, for each new disjunctive node that generates an output value, a new outgoing edge from the new disjunctive node to a corresponding conjunctive node.

20. A method according to claim 19 wherein each outgoing edge has an origin and a destination and only responds to a true value at the origin.

21. A method according to claim 19 wherein each outgoing edge has an origin and a destination and only responds to a false value at the origin.

22. A method according to claim 19 wherein each outgoing edge has an origin and a destination and only asserts a false value at the destination.

23. A method according to claim 19 wherein each outgoing edge has an origin and a destination and only asserts a true value at the destination.

24. A method according to claim 19 wherein each outgoing edge has an origin in which the edge responds to a value selected from the group consisting of true and false, at the origin.

25. A method according to claim 24 wherein each outgoing edge has a destination in which the edge asserts a value selected from the group consisting of true and false, at the destination.